МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»**

Кафедра математического и компьютерного моделирования

Разработка информационной системы учета публикационной активности преподавателей ВУЗа

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 441 группы

направления 09.03.03 «Прикладная информатика в экономике»

механико-математического факультета
Россошанского Георгия Васильевича

Научный руководитель _____ И.В. Плаксина
доцент, д.ф.-м.н.,

Зав. кафедрой д.ф.-м.н., _____ Ю.А. Блинков
доцент

Саратов 2020

ВВЕДЕНИЕ

Уже почти пятьдесят лет, с момента его появления и по сей день в Интернете стабильно отслеживается положительная динамика роста пользователей: на конец 2018 года их общее количество в мире достигло 4,2 млрд человек.

Увеличению популярности Глобальной сети сопутствует ряд причин, а конкретно: внедрение новых технологий, повышающих мощность ресурсов ПК и серверов, а также пропускную способность маршрутизаторов; минимальные затраты на создание сайта: как финансовые, так и интеллектуальные (в том числе низкая себестоимость телекоммуникационного оборудования).

Эволюция Web 2.0 дала толчок к увеличению массы контента в сети, генерируемого обычным пользователем. Соцсети, журналы, интернет-магазины ежегодно пропускают сквозь себя огромные потоки информации и данных и как результат – появление в обществе глобального тренда, именуемого информационным прорывом (ИП). При разработке софта, использующего колоссальные объемы информации, приходится решать две основные проблемы, вытекающие из ИП: обработка экспоненциально нарастающих массивов данных в реальном времени; разработка решений, направленных на существенное сокращение времени анализа данных.

В связи с этим своевременным становится проектирование алгоритмов и методов получения нужной информации из неструктурированных данных Интернета. Данным явлением обусловлено мощное развитие новейших алгоритмов последовательного синтаксического анализа данных, опубликованных на интернет-ресурсах – парсинга сайтов.

Также встает вопрос о необходимости увеличения гибкости и универсальности используемых структур данных. Повышается спрос в области анализа данных различного рода, как структурированных, так и не структурированных, зачастую требуется быстро модифицировать не только данные, но и их структуру и взаимосвязи. Во многих существующих и новых информационных технологиях все больше преобладает «четвертая парадигма». Ставшие уже классическими для задач хранения и обработки информации SQL-базы данных не всегда отвечают предъявляемым требованиям. С каждым днем

все большее применение находят различные NoSQL-решения. Причин этому несколько:

- резкий рост объемов обрабатываемых и хранимых данных;
- сложная и изменчивая структура данных;
- высокие требования к возможности распределенного хранения данных;
- нестандартные методы обработки данных.

Одним из самых перспективных направлений развития NoSQL-решений являются графовые базы данных. Можно выделить следующие преимущества данных решений:

- разработка без задания схемы данных;
- наглядность;
- легкая масштабируемость;
- широкие возможности по описанию сложных данных;
- описание данных с большим количеством связей.

Данная дипломная работа посвящена разработке информационной системы учета публикационной активности преподавателей ВУЗа. Цель работы – исследование проблемы скрапинга в контексте наполнения нереляционной базы данных для хранения и обработки информации о научных публикациях различного типа, изучение применения его методов и случаев невозможности его применения, а также знакомство с инструментарием языка программирования Python для практического решения задач, связанных с парсингом, решение проблемы хранения данных, изучение применения методов хранения данных, а также знакомство с языком для написания запросов к базе данных Neo4j Cypher.

К задачам работы относятся: изучение основных понятий концепции скрапинга и написание алгоритма для наполнения нереляционной базы данных на языке программирования Python, извлекающего информацию с сайта национальной библиографической базы данных научного цитирования (РИНЦ), изучение основных понятий концепции графовой базы данных, знакомство с СУБД Neo4j, проектирование модели графовой БД.

Дипломная работа состоит из введения, пяти разделов, кода информационной системы, заключения, списка использованных источников и приложения. Во введении содержится краткое описание проблемы, приведены це-

ли и задачи работы. В первом разделе формулируются основные понятия, необходимые для дальнейшей работы и оперирования терминами, а также принцип скрапинга. Во втором разделе рассматривается пример практического использования парсера. В третьем разделе рассматривается концепция NoSql графовой базы данных. В четвертом разделе проектируется модель хранения данных для будущего приложения. В пятом разделе описывается процесс разработки алгоритма, позволяющего объединить предыдущие шаги и завершить приложение. В заключении приведены основные результаты и выводы о проделанной работе. Список использованных источников содержит 20 наименований.

1 Основные определения, используемые в дипломной работе

Определение 1. Web Scraping - процесс получения данных с web-сайтов, используя программные средства.

Определение 2. Парсер (Parser) - часть программы, преобразующей входные данные (как правило, текст) в структурированный формат. Парсер выполняет синтаксический анализ текста.

Определение 3. Парсинг (Parsing) - процесс преобразования входных данных в структурированный формат, пригодный для синтаксического анализа.

Определение 4. Graph database (графовая база данных) — база данных построенная на графах — узлах и связях между ними.

Определение 5. Cypher — язык для написания запросов к базе данных Neo4j (примерно, как SQL в MYSQL).

Определение 6. Node (нода) — объект в базе данных, узел графа. Количество узлов ограниченно 2 в степени 35 34 биллиона.

Определение 7. Node label (метка ноды) — используется как условный «тип ноды». Например, ноды типа movie могут быть связанны с нодами типа actor. Метки нод — регистрозависимые, причем Cypher не выдает ошибок, если набрать не в том регистре название.

Определение 8. Relation (связь) — связь между двумя нодами, ребро графа. Количество связей ограниченно 2 в степени 35 34 биллиона.

Определение 9. Relation identirfier (тип связи) — в Neo4j у связей. Максимальное количество типов связей 32767.

Определение 10. Properties (свойства ноды) — набор данных, которые можно назначить ноде. Например, если нода — это товар, то в свойствах ноды можно хранить id товара из базы MySQL.

Определение 11. Node ID (ID нода) — уникальный идентификатор ноды. По умолчанию, при просмотрах результата отображается именно этот ID.

2 Анализ современных методов организации веб парсинга

2.1 Начальные понятия и подходы

В общем значении Web Scraping – это получение данных программным методом с различных интернет ресурсов. Суть его работы возможно определить следующим способом: программный код выполняет GET-запрос на выбранный веб-ресурс, получает отклик, проводит синтаксический анализ HTML документа и с помощью внутренних библиотечных методов навигации и синтаксического анализа преобразует данные в заданный формат. Обратим внимание на то, что к классу полезных данных могут относиться:

- товарные каталоги;
- фото контент;
- видео;
- текстовый контент (в том числе и открытые контактные данные: адреса электронной почты, телефоны и т.д.);

Есть масса решений для Web Scraping, такие как:

- Отдельные сервисы, которые работают через API или имеют веб-интерфейс (Embedly, DiffBot и др.).
- Проекты с открытым кодом, на разных языках программирования (Goose, Scrapy — Python; Goutte — PHP; Readability, Morph — Ruby).
- Собственное решение (например, с использованием библиотеки Nokogiri (для языка программирования Ruby)).

2.2 Проблемы, возникающие на практике, и их решение

Невозможно разработать универсальный парсер, который бы мог получить информацию с любого веб-ресурса. Рассмотрим несколько проблем, которые возникают при попытке создать такую программу для всех типов сайтов.

1. Каждый сайт имеет свою собственную верстку в масштабах всего интернета, что и делает возможным конкуренцию в поисковой выдаче за

звание лучшего веб-ресурса. Не существует идеального решения с точки зрения догматов веб-дизайна.

2. Существует проблема «самописного» кода, ведь не каждый веб-разрабочик придерживается общих правил и инструкций, а зачастую пишет код под себя или просто, как умеет. Такой код обычно не получается грамотным и качественным, так как в нем можно найти бесчисленное количество ошибок. Все это делает такой код полностью нечитаемым для скраперов (речь, в первую очередь, идет о верстке).
3. Множество веб-сайтов использует HTML5, где все элементы могут быть абсолютно уникальными между собой.
4. Некоторые ресурсы содержат разнообразные защиты от копирования данных, а значит и от скрапинга. Это выражается в многоуровневой верстке, использовании JavaScript для рендеринга контента, проверки user-agent и т.д. Так же внедряются особые страницы недоступные обычному пользователю, но доступные для парсеров, что и является ловушкой, которая при заходе на подобную страницу блокирует дальнейшую возможность парсинга.
5. В зависимости от сезона или тематики целевого материала на сайте могут быть использованы разные макеты. Периодически это касается даже типичных страниц (сезонные акции, премиум статьи и т.д.).
6. Кроме полезных блоков, веб-страница часто изобилует «мусором» в виде рекламы, комментариев, дополнительных элементов навигации и т.д.
7. Исходный код может содержать ссылки на одни и те же картинки разных размеров, например — для превью.
8. У всех сайтов может быть разная кодировка, которая не определяется в ответе на запрос.

Описанные проблемы сильно утяжеляют процесс веб-скрапинга. В итоге качество полученного контента может снизиться, что абсолютно недопустимо, т.к. полнота информации является важнейшим критерием качества. Для того, чтобы избежать подобных ситуаций необходимо выполнение следующих правил:

- В случае, если данные необходимо получить из ограниченного количества источников, предпочтительно написать собственный парсер и настроить его под нужные сайты.
- Если нужно получить информацию из большого количества источников, то необходимо использовать комплексный подход в выборе парсера.

3 Эффективное хранение данных

3.1 Проблематика хранения данных

На сегодняшний день из всех существующих технологий и инструментов на рынке представлено очень малое количество тех, которые позволяли бы с большой гибкостью описывать объекты реального мира, хранить полученные данные и выполнять их обработку, анализ и изменение самих данных, не выходя за рамки изначально используемой базы данных.

Таким образом, при решении определенного круга задач, например, таких как всевозможные научные исследования, описание и отслеживание бизнес-логики и бизнес-процессов, существует острая потребность в некотором хранилище, которое позволяло бы при помощи него осуществлять полный цикл сбора, хранения, обработки данных в единой адаптирующейся модели. Это могло бы дать значительный выигрыш не только в скорости доступа к данным и их обработке, но и в удобстве использования, упрощении и ускорении процесса разработки и интеграции, снижении избыточности данных и количестве применяемых инструментов, а соответственно и затрат на поддержку информационной системы в будущем.

Подобные многоцелевые решения пригодятся в первую очередь в сфере анализа данных, динамически изменяющихся в процессе работы с ними. Это те случаи, где можно столкнуться с ограничениями и неудобствами условно стандартных и распространенных средств хранения, таких как реляционные базы данных или многие NoSQL-решения, предоставляющие в большинстве случаев лишь широкие возможности по хранению и быстрому доступу к данным на физическом уровне.

Проблема заключается не только в скорости доступа к хранимым данным, но также и в правильной их логической организации посредством минимизации стоимости каждой операции, совершаемой над данными. Одним из центральных методов представления данных об окружающем мире в информационных системах является метод, основанный на выделении из доступного объема информации метаданных и непосредственно самих данных.

Суть данного подхода состоит в некотором анализе всего множества описываемых объектов, выделении на его основе некоторого количества различ-

ных классов и описании реальных объектов при помощи них. Каждый выделенный класс обладает определенным набором свойств, присущих ему.

В реальном же мире зачастую невозможно сразу определить класс объекта, либо объекты, отнесенные к одному классу, будут обладать различным набором свойств. Нет ничего идеального и поддающегося строгому описанию. Такой подход применим лишь для определенного круга задач, и способен отражать суть только статически описываемых объектов.

Перспективной моделью данных, подходящей под вышеперечисленные критерии и лишенной необходимости предварительной классификации описываемых объектов, является графовая модель данных, достаточно гибкая для описания данных любого рода и сложности, в силу своих структурных особенностей. Графы одинаково хорошо подходят для представления как слабо, так и для сложно структурированных объектов и систем. Помимо всего прочего графовую модель данных возможно представить как некоторый рекурсивный тип данных, что дает широкие возможности по применению различных способов его хранения и обработки. Здесь открывается целый спектр методов и алгоритмов для работы как с самой структурой графа и его логической организацией, так и непосредственно с данными, хранящимися в нем.

Области применения графовых моделей и методы работы с ними активно изучаются, ведется множество исследований, направленных на унификацию языков запросов к базам данных, выбор канонических моделей и моделей ресурсов. Такие исследования, безусловно, необходимы и играют одну из ключевых ролей.

Неотъемлемой частью успешной обработки данных на графовых моделях являются способы хранения и методы доступа к данным внутри используемых структур. Обычно способы хранения графов в памяти обладают высокой избыточностью и лишены возможности быстрого обхода и поиска, что приводит к сильному снижению скорости обработки. Таким образом, сейчас стоит задача по пересмотру и анализу существующих способов представления и хранения графовых моделей в памяти компьютера, выявление их особенностей, преимуществ и недостатков. Необходимо сконцентрировать силы на поиске нового, гибкого, легко адаптируемого под поставленные задачи способа

представления графов. Искомая структура данных должна будет способствовать оптимизации используемой памяти логической организации, тем самым позволив снизить избыточность, ускорить процесс индексации и поиска хранимых данных, понизив алгоритмическую сложность обработки.

3.2 О базах данных

Сегодня данные, которые необходимо где-то хранить – это весьма непредсказуемая структура, которая со временем может превратиться либо в BigData, либо в сложную семантическую сеть, и часто разработчик не может заранее сказать, какой она будет. Возникает необходимость выбора базы данных – или хотя бы ее архитектуры, чтобы создать действительно быстрое и эффективно работающее приложение.

Как известно, существует несколько видов баз данных. Первый и самый известный – это реляционные базы данных с их единым языком SQL. Именно благодаря стандартизации реляционные базы данных заработали себе популярность и доминируют на рынке. Но по факту реляционные базы данных – это просто таблицы, где в каждой строке выстраивается однозначное соответствие между ключом и его многочисленными (или малочисленными) параметрами. Пока приложения ограничивались отдельными таблицами и не рассматривали особенных взаимодействий между собой и разными типами данных, этого было вполне достаточно.

Как альтернатива базам данных SQL с начала 2000-х развивается направление NoSQL. В эту категорию объединяют базы данных, начиная от иерархических и сетевых БД (где помимо иерархии предусмотрены дополнительные связи) до упрощенных БД ключ-значение и документарных баз данных без определенных параметров значений каждого элемента. Причина эволюции этой категории баз данных заключается в следующем: если рассматривать примитивные и однотипные наборы данных, а запросы касаются одной таблицы – можно работать с SQL. Но если возникает ситуация, когда нужно обратиться к 10, 100, 1000 и тд. таблицам, чтобы обработать запрос, реляционная база данных начинает работать медленно, а для написания запроса потребуется создать код немалого объёма.

Наибольшей популярностью баз данных из категории NoSQL пользуются документарные БД, в частности, MongoDB. Они позволяют хранить объекты с произвольными наборами значений.

Отдельным классом стоят графовые базы данных. Они предполагают более естественное представление информации. Каждая социальная сеть — это граф, и сетевая модель БД – это тоже фактически граф, но без дополнительных возможностей, которые открывает современная графовая модель. Поэтому графовые базы данных представляют особый интерес для разработчиков.

3.3 Плюсы и минусы различных типов БД

Итак, реляционная архитектура – это хорошее решение для тех случаев, когда всё просто и однозначно, но совершенно неудобная архитектура для создания сложных и гибких запросов, обработки разнообразных и многократных связей между объектами. Однако, нельзя забывать о таких преимуществах SQL-баз данных, как возможность создания сложных (JOIN) запросов. Такой подход делает стандартизированные реляционные БД более универсальными, пусть даже с большим количеством кода, но каждый запрос может быть в них реализован. Например, запрос «найти всех людей моложе 20 лет, у которых есть автомобили красного цвета», будет достаточно легко реализовать в SQL, в то время как БД из категории NoSQL потребуют массы усилий для решения этой задачи.

Прямая альтернатива SQL – документарные базы данных. Их главным преимуществом является отсутствие единой схемы всех элементов (schemaless). В отличие от SQL эти базы могут сохранять любой сложный объект, например, документ с большим количеством полей за одну операцию, а также за одну операцию выдавать его. Это очень удобно, например, для добавления новых категорий товаров в каталог интернет-магазина, ведь для каждого товара будут применяться совершенно разные свойства. В MongoDB можно работать с ними через короткие запросы, в то время как в SQL для получения и обновления такой сложной записи придется создавать специальные процедуры, выполняющие множество запросов.

Минусы документарных баз также вытекают из их архитектурных особенностей. Например, документарная модель не подразумевает таких простых функций объединения (JOIN), а также возможности работать с двунаправленными связями. Кроме этого документарная база рассчитана на хранение отдельных элементов, не имеющих дополнительных связей между собой.

3.4 Подробнее о графовой БД

Графовая база данных – разновидность баз данных с реализацией сетевой модели в виде графа и его обобщений. Такие базы данных, в первую очередь, предназначены для решения тех задач, где данные, тесно связанные между собой в отношениях, которые могут углубляться в несколько уровней. Например, в реляционных базах данных не трудно выполнить запрос: «выдать список всех актеров, которые были в фильме с Кевином Бэконом». Но если возникает необходимость получить «имена всех актеров, которые были в кино с кем-то, кто был в кино с Кевином Бэконом», появляется ещё один JOIN. Если теперь добавить третью степень: «Тот, кто был в кино с кем-то, кто был в кино с кем-то, кто был в фильме с Кевином Бэконом». Задача реальная, и с каждой новой связью необходимо добавлять JOIN, а запрос будет становится все более сложным, трудоёмким, всё менее производительным.

Глубокие связи особенно актуальны в различных социальных проектах, когда нужно получать друзей друзей, в задачах поиска маршрутов и т.п. Графовые базы данных призваны решить эти проблемы, когда данные могут быть удалены друг от друга на два и более отношений. Они решаются применением моделирования данных как «вершин графов», а связей как «ребра графа» между этими узлами. Можно так же делать обход графа с помощью давно известных и эффективных алгоритмов.

Основным преимуществом графовых баз данных в этом свете является универсальность, так как в них можно хранить и реляционные, и документарные, и сложные семантические данные. Сама же модель построения БД может меняться и модифицироваться в процессе развития приложения без изменения архитектуры и исходных запросов.

С другой стороны, при незначительном количестве связей и больших объемах данных графовые БД имеют значительно более низкую производительность. Еще одним важным ограничением является то, что в данный момент практически не существует графовых баз данных, которые бы хорошо работали в параллельных архитектурах.

3.5 Графы все же перспективны?

Однако, если говорить о разработке приложений, в процессе проектирования и даже на стадии обработки нередко появляются новые требования к структуре данных. Добавление новых связей делает неприемлемой документарную базу данных, а рост количества JOIN-ов катастрофически снижает производительность реляционной БД. В этом случае графы оказываются наиболее универсальным вариантом, позволяющим подстраховаться на случай изменения требований и расширения функционала в будущем.

Более того, сегодня активно идет доработка RDF – основного стандарта, согласно которому работают графовые базы данных. Именно стандартизация SQL сделала такими популярными реляционные БД. При этом ряд проектов демонстрирует поддержку OData для создания стандартных веб-запросов через HTTP, а также язык SPARQL, обладающий широкими возможностями для работы с различными видами запросов и данных. За счёт развития архитектуры производительность графовых БД растет, и, возможно, скоро окажется выше реляционной даже при небольшом количестве связей.

ЗАКЛЮЧЕНИЕ

Сегодня остро стоит проблема выбора инструментов для сбора и обработки огромного количества данных, а также средств её хранения: стало важно не просто получить требуемую информацию, а сделать это эффективно, то есть с минимальными временными и технологическими затратами, а как следствие и экономическими.

Интернет-магазины, онлайн-библиотеки, новостные агрегаторы и т.д. стремятся оптимизировать свои сайты и базы данных таким образом, чтобы конечный пользователь мог быстро получить искомую информацию. Поскольку пользователь не задается вопросом какие ресурсы привлекаются для того, чтобы удовлетворить его поисковую потребность, ведется ожесточенная конкурентная борьба за конкурентное преимущество в тысячные доли отклика, чтобы в конечном итоге замедлить экспоненциальный рост массивов данных.

Для этого производится анализ новейших систем обработки и хранения данных, разрабатываются новые языки программирования для ускорения взаимодействия с базами данных.

В данной работе была сформулирована проблемы скрапинга в контексте наполнения нереляционной базы данных для хранения и обработки информации о научных публикациях, изучено применение его методов и случаев невозможности его применения, сформулирована проблема хранения данных в контексте наполнения нереляционной базы данных научными публикациями различного типа, изучены применения методов хранения данных, а также было проведено знакомство с языком для написания запросов к базе данных Neo4j Cypher,как и с инструментарием языка программирования Python для практического решения задач, связанных с парсингом.

Были изучены основные понятия концепции скрапинга, понятия концепции графовой базы данных, изучен основной функционал СУБД Neo4j, проанализирован инструментарий выбранного языка программирования, подходящий для этих целей, а также непосредственно информационная область, и написан алгоритм для наполнения нереляционной базы данных на языке программирования Python.